CLAIMS

1. [Cancelled]

2. [Currently Amended] The method of claim ~~1~~ 34, wherein each of the one or more data sequence has a length ~~is a~~ of one byte.

3. [Currently Amended] The method of claim ~~1~~ 34, wherein each of the one or more data sequence has a length ~~is~~ of one~~a~~ bit.

4. [Currently Amended] The method of claim ~~1~~34, wherein the the instruction received from the processor ~~computing system comprises a processor having~~is a member of an extensible instruction set.

5. [Currently Amended] The method of claim ~~1~~34, wherein each of the one or more unaligned data sequence has a length less than a length of the second aligned word~~the computing system comprises a general purpose processor~~.

6. [Currently Amended] The method of claim ~~1~~34, ~~wherein the first aligned word is stored in a first memory location and the second aligned word is stored in an adjacent second memory location, and the second memory location is accessed by incrementing a memory address pointer after the first aligned word is accessed~~further including changing

a memory address pointer by an amount less than a length of the second aligned word to point to a next unaligned data sequence to be read.

7-8. [Cancelled]

9. [Currently Amended] The method of claim ~~8~~ 35, wherein ~~the~~ a length of each of the one or more unaligned data sequences is one~~data sequence length is a~~ byte.

10. [Currently Amended] The method of claim ~~8~~35, wherein a length of each of the one or more unaligned data sequences is one~~the data sequence length is a~~ bit.

11. [Currently Amended] The method of claim ~~8~~35, wherein the ~~computing system comprises a processor having~~instruction received from the processor is a member of an extensible instruction set.

12. [Currently Amended] The method of claim ~~8~~35, further including flushing the load/store buffer in order to write any remaining unaligned data into the memory~~wherein the computing system comprises a general-purpose processor~~.

13. [Currently Amended] The method of claim ~~8~~35, wherein a length of each of the one or more data sequence is less than a length of an aligned word in the memory~~wherein the aligned word is stored in a first memory location and an next aligned word is stored in an~~

~~adjacent second memory location, and the second memory location is accessed by incrementing a memory address pointer after the first aligned word is accessed~~.

14-16. [Cancelled]

17. [Currently Amended] A system comprising:

a load/store buffer configured to store data; and

a processor configured to execute GET instructions for processing data sequences, the GET instructions comprising the steps:

initializing the load/store buffer by loading a first aligned word ~~of fixed length~~ into the load/store buffer from memory;

further initializing the load/store buffer by loading a second aligned word from the memory into the load/store buffer;

reading one or more data sequences from the load/store buffer into a register file configured for instruction execution using an instruction received from the processor, at least one of the one or more data sequences being unaligned relative to the memory~~such that the total length of the sequences in each read does not exceed the fixed length of the first aligned word~~; and

loading additional aligned words to the load/store buffer from the memory to replace the one or more data sequences that are read from the load/store buffer into the register file.

18. [Previously Presented] The system of claim 17 in which the number of data sequences read is an immediate specified number.

19. [Previously Presented] The system of claim 17 in which the number of data sequences read is a specified number stored as an index in a register memory.

20. [Currently Amended] The system of claim 17 in which a first of the one or more data sequences read is located at a first memory location and the one or more data sequences comprises a specified number of data sequences stored as a first index in a register memory, wherein ~~the~~ a subsequent data sequence following ~~the~~ a first of the data sequences is located at a second memory location pointed to by a second index.

21. [Cancelled]

22. [Currently Amended] The system of claim ~~21~~ 38 in which ~~the~~ a number of the one or more unaligned data sequences~~unaligned data sequences~~ written is an immediate specified number.

23. [Currently Amended] The system of claim ~~21~~38 in which ~~the~~ a number of ~~unaligned data sequences~~the one or more unaligned data sequences written is a specified number stored as an index in a register memory.

24. [New] A method of processing data, the method comprising:

receiving a sequence of aligned data at an extension adapter, the receipt of the sequence controlled by a processor using a load/store instruction;

receiving a user defined instruction at the extension adapter from the processor;

loading an unaligned subset of the sequence into a programmable instruction set extension fabric; and

executing the user defined instruction using the unaligned subset as an operand.

25. [New] The method of claim 24, further including stalling execution of the user defined instruction to assure availability of the unaligned subset.

26. [New] The method of claim 24, further including controlling the timing of the loading of the unaligned subset as a function of instruction executiontime.

27. [New] The method of claim 24, further including operating on both aligned and unaligned operands using the programmable instruction set extension fabric, the alignment being relative to a word size of the processor.

28. [New] The method of claim 24, wherein loading the unaligned subset of the sequence into the programmable instruction set extension fabric is accomplished using an instruction configured to read data of a first size and to increment a memory address pointer by an amount different from the first size.

29. [New] The method of claim 24, further including initializing a data pointer in order to achieve an offset between a data alignment of the processor and the unaligned subset.

30. [New] The method of claim 24, wherein loading the unaligned subset of the sequence is performed using a PUT instruction received from the processor.

31. [New] A method of processing data, the method comprising:

receiving a sequence of aligned data, the aligned data being aligned relative to a memory;

receiving a user defined instruction;

loading an unaligned subset of the sequence of aligned data into a programmable instruction set extension fabric;

executing the user defined instruction using the unaligned subset as an operand to create an unaligned instruction output;

receiving the unaligned instruction output;

aligning unaligned instruction output to the memory to create aligned data; and

storing the aligned data in the memory.

32. [New] The method of claim 31, wherein receiving the sequence of aligned data is controlled by a processor using a load/store instruction, and the user defined instruction is received from the same processor.

33. [New] The method of claim 31, wherein sequence of aligned data is received from a first location within the memory and the aligned data is written to a second different location within the memory.

34. [New] A method for processing data sequences in a computing system, the method comprising:

initializing a load/store buffer by loading a first aligned word into a load/store buffer;

further initializing the load/store buffer by loading a second aligned word into the load/store buffer, alignment of the first aligned word and the second aligned word being relative to a memory accessible via a processor;

reading one or more unaligned data sequence from the load/store buffer into a register file for use by an instruction received from the processor, the unaligned data sequence including at least part of the second aligned word; and

loading additional aligned words to the load/store buffer to replace the first aligned word and the second aligned word.

35. [New] A method for processing data sequences in a computing system, the method comprising:

generating one or more unaligned data sequences using an instruction received from a processor;

initializing a load/store buffer by loading data to the load/store buffer,;

writing the one or more unaligned data sequence to the initialized load/store buffer; and

writing the one or more unaligned data sequences from the initialized load/store buffer to a memory, using a load/store instruction from the processor, such that the one or more unaligned data sequence becomes aligned with the memory.

36. [New] A system for processing data sequences in a computing system, the system comprising:

means for initializing a load/store buffer by loading one or more unaligned data sequences into the load/store buffer, a length of each of the one or more unaligned data sequence being less than a length of an aligned word of a memory, the one or more unaligned data structures being unaligned relative to the memory and including output of a user defined instruction executed using data retrieved from the memory; and

means for writing the one or more unaligned data sequences to the memory, such that the written unaligned data becomes aligned to one or more word boundary of the memory.

37. [New] The system of claim 17, wherein a length of at least one of the one or more data sequences read from the load/store buffer into the register file is less than a length of the first aligned word.

38. [New] A system comprising:

a load/store buffer configured to store data; and

a processor configured to execute PUT instructions for processing data sequences in a computing system, the PUT instructions comprising:

initializing the load/store buffer by filling the load/store buffer with one or more unaligned data sequences from a register file a length of each of the one or more unaligned data sequence being less than a length of an aligned word, the one or more unaligned data sequences being generated using part of the aligned word; and

writing one or more unaligned data sequences from the initialized load/store buffer, such that the unaligned data sequences become aligned relative to the aligned word.